Contenido

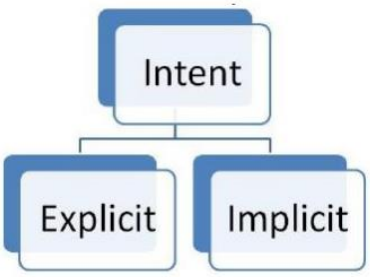
5.

Enlazando Activity's

INTRODUCCIÓN

En Android podríamos definir los intents o intenciones como la voluntad de realizar una acción o una tarea determinada, estas tareas pueden ser una llamada de teléfono o visualizar una página web entre otras. Básicamente un intent nos permite lanzar una actividad o servicio de nuestra aplicación o de una aplicación diferente. Pertenece al paquete "android.content".

Android soporta dos tipos de Intents: explícitos e implícitos.



INTENT EXPLÍCITO

Se especifica exactamente el componente a lanzar. Se suele utilizar cuando se quiere ejecutar los componentes internos de una aplicación.

En el inicio de una nueva actividad se pueden dar varias circunstancias:

- Inicio de una actividad sin argumentos ni devolución de resultados:

```
Intent intento = new Intent(this, clase.class);
```

o

```
Intent intento = new Intent(getApplicationContext(), clase.class);
```

Y

```
startActivity(intento);
```

- Inicio de una actividad con argumentos y sin devolución de resultados. Podemos pasar información a las actividades añadiendo datos al objeto intent con el método putExtra:

```
Intent intento = new Intent(this, clase.class);
intento.putExtra(atributo, valor);
startActivity(intento);
```

  En la clase invocada obtendremos los datos recibitos de la siguiente manera:

```
Intent intento = getIntent();
String variable = intento.getStringExtra(atributo, valorDefecto);
…
```

  O también todos de golpe a través de *getExtras()*, que nos devuelve un bundle, al que podemos solicitar los extras a través del atributo.

- Inicio de una actividad con argumentos y devolución de resultados. Podemos pasar información a las actividades añadiendo datos al objeto intent con el método putExtra y obtener los resultados con el método onActivityResult:

```
Intent intento = new Intent(this, clase.class);
intento.putExtra(atributo, valor);
startActivityForResult(intento,cod);
…
public void onActivityResult (int codActividad, int codResultado, Intent datos) {
        if (codResultado == RESULT_OK) {
                …
        }
        …
}
```

En la clase invocada devolveremos los resultados de la siguiente manera:

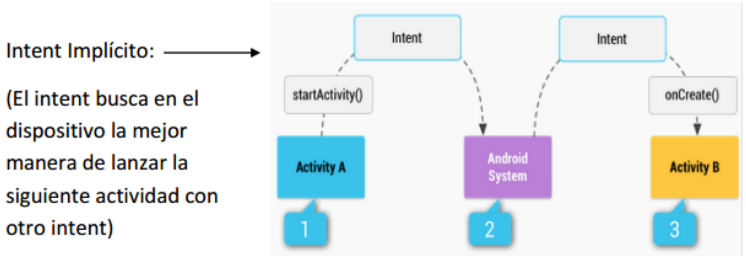
```
Intent datos = new Intent();
datos.putExtra(atributo,valor);
setResult(RESULT_OK,datos);
finish();
```

- **Ejercicio Resuelto2Activitys**
- **Ejercicio Propuesto3ActivitysRetornoDatos**

INTENT IMPLÍCITO

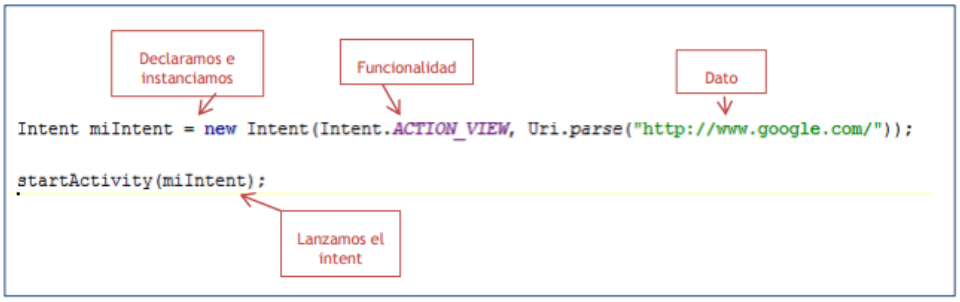
Los Intent implícitos son aquellos que le preguntan al sistema qué servicio o componente es el más adecuado para realizar la petición, es decir, no específica un componente en especial, y solo se especifica la funcionalidad requerida a través de una acción (ver, hacer búsqueda, hacer foto, marcar número…) y un dato (URL, número a marcar…). En este caso el sistema

determina el componente para su utilización, desencadenando un "intent" sin conocer exactamente cuál es la aplicación o componente que lo recibirá, si no puede establecer exactamente el componente a utilizar porque hay más de uno muestra una ventana con las opciones a seleccionar.



Básicamente los pasos para lanzar una actividad con un "intent implícito" serían los siguientes:

- Declaramos el "intent" con la acción apropiada (ACTION_VIEW, ACTION_WEB_SEARCH, etc).
- Adjuntamos información adicional necesaria dependiendo de la acción del intent.
- Lanzamos el intent dejando que el sistema encuentre la actividad adecuada).



Para iniciar la actividad utilizaremos igualmente startActivity.

Algunos ejemplos:

- Para mostrar una web dentro de un navegador:

```
Intent intento = new Intent(Intent.ACTION_VIEW,
Uri.parse("http://www.google.es"));
startActivity(intento);
```

  Requiere los permisos:

```
<uses-permission android:name=" android.permission.INTERNET "/>
```

- Para utilizar la búsqueda de google;

```
Intent intento = new Intent(Intent.ACTION_WEB_SEARCH);
intento.putExtra(SearchManager.QUERY, "I.E.S. Castelar");
startActivity(intento);
```

Requiere los permisos:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

- Para mostrar los contactos.

```
Intent intento = new Intent(Intent.ACTION_VIEW,
        Uri.parse("content://contacts/people/"));
startActivity(intento);
```

Requiere los permisos:

```
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```

- Para hacer una llamada una vez pulsado el botón llamar:

```
Intent intento = new Intent(Intent.ACTION_VIEW,
        Uri.parse("tel:123456789"));
startActivity(intento);
```

Requiere los permisos:

```
<uses-permission android:name="android.permission.CALL_PHONE" />
```

- Para hacer una llamada automática con el dial:

```
Intent intento = new Intent(Intent.ACTION_DIAL,
        Uri.parse("tel:123456789"));
startActivity(intento);
```

Requiere los permisos:

```
<uses-permission android:name="android.permission.CALL_PHONE" />
```

- Para mostrar un mapa:

```
Intent intento = new Intent(Intent.ACTION_VIEW, Uri.parse("geo:38.879398,
    -6.9781"));
startActivity(intento);
```

Requiere los permisos:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

- Para hacer una foto.

```
Intent intento = new Intent("android.media.action.IMAGE_CAPTURE");
startActivity(intento);
```

Requiere los permisos:

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />
```

- **EjercicioResueltoIntentImplicito**
- **EjercicioPropuestoIntentImplicito**

INTENT-FILTER

Las actividades y servicios pueden declarar el tipo de acciones que pueden llevar a cabo y los tipos de datos que pueden gestionar. De esta forma se informa al sistema del tipo de intenciones implícitas que puede atender dicho componente.

Los intent-filter se definen en el archivo de manifiesto de la aplicación dentro de la activity correspondiente.

Cuando se crea un intent inplícito, el sistema Android busca el componente apropiado que pueda resolver la petición dada en el intent. La búsqueda la realiza analizando los intent.filter definidos en los archivos de manifiesto de las aplicaciones instaladas en el dispositivo. Si existe coincidencia lanza dicha aplicación. Si la coincidencia es múltiple (varias aplicaciones pueden dar respuesta a esa petición), el sistema muestra un cuadro de diálogo con el fin de que el usuario decida.

A un Intent podemos asociarle una **acción**, unos **datos** y una **categoría**. Las actividades pueden declarar el tipo de acciones que pueden llevar a cabo y los tipos de datos que pueden gestionar. Las acciones son cadenas de texto estándar que describen lo que la actividad puede hacer. Por ejemplo *android.intent.action.VIEW,* es una acción que indica que la actividad puede mostrar datos al usuario. La misma actividad puede declarar el tipo de datos del que se ocupa, por ejemplo *vnd.android.cursor.dir/person*, indica que la actividad manipula los datos de la agenda. También pueden declarar una categoría, que básicamente indica si la actividad va a ser lanzada desde el lanzador de aplicaciones, desde el menú de otra aplicación o directamente desde otra actividad. En el AndroidManifest.xml quedaría algo así:

```
<intent-filter>
<action android:name="android.intent.action.VIEW" />
<category android:name="android.intent.category.DEFAULT" />
<data android:mimeType="vnd.android.cursor.dir/person" />
</intent-filter>
```

Una activity puede tener varios intent-filter definidos.

Igualmente cada uno de esos intent-filter puede tener varias acciones definidas.

```
<activity  class = ".NotesList"  android:label = "@string/title_notes_list" >
    <intent-filter>
        <action  android:name = "android.intent.action.MAIN"  />
        <category  android:name = "android.intent.category.LAUNCHER"  />
    </intent-filter>
    <intent-filter>
        <action  android:name = "android.intent.action.VIEW"  />
        <action  android:name = "android.intent.action.EDIT"  />
        <action  android:name = "android.intent.action.PICK"  />
        <category  android:name = "android.intent.category.DEFAULT"  />
        <data  android:mimeType = "vnd.android.cursor.dir/ vnd.google.note "  />
    </intent-filter>
    <intent-filter>
        <action  android:name = "android.intent.action.GET_CONTENT"  />
        <category  android:name = "android.intent.category.DEFAULT"  />
        <data  android:mimeType = "vnd.android.cursor.item/ vnd.google.note "  />
    </intent-filter>
</activity>
```

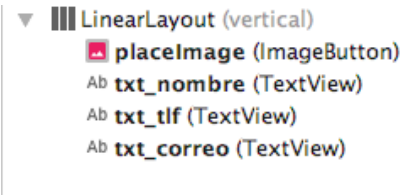
El primer intent-filter define que esta activity es punto de entrada para la aplicación y la categoría especifica que debe aparecer en el lanzador de aplicaciones.
El segundo, permite a la activity visualizar y editar el directorio de datos (a través de las actions VIEW y EDIT), o recuperar una entrada particular y devolverla (a través de la action PICK).
En data se especifica el tipo de datos que se manejan. Como está definido con la Categoría DEFAULT, estamos indicando que atenderá a los intents implícitos que soliciten otras aplicaciones.

TRABAJANDO CON PARCELABLES

Cuando estamos desarrollando en Android y necesitamos enviar datos a otra Activity lo podemos realizar mediante un **Intent y Bundle** pero qué sucede si deseamos enviar un objeto para economizar una llamada a la API. Android nos brinda las opciones de **Serializable** (propia de Java) y **Parcelable,** sin embargo esta última es hasta 10 veces más rápida por lo que nos vamos a enfocar en ella.

Parcelable es una interfaz y podemos implementarla manual o mediante un plugin, usaremos el plugin que nos facilita el trabajo+. Claramente tendremos que crear la clase POJO, como se hace siempre pero está deberá derivar de la Interfaz Parcelable. Como es de suponer esto nos obligará a implementar una serie de métodos y una variable **Parcelable.Creator**.

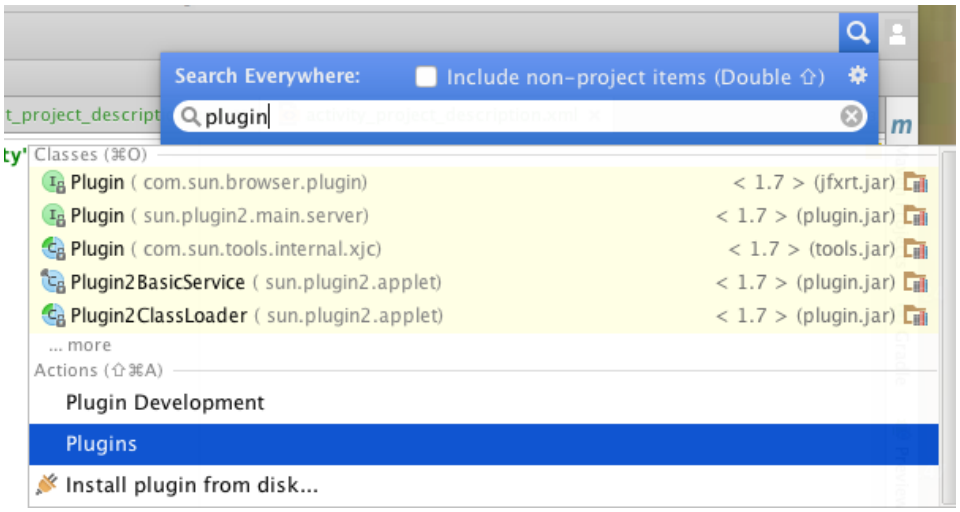
Veamos un ejemplo tomando como base el proyecto del ejercicio propuesto de intents implícitos. Incluiremos en la interfaz un botón que envíe los datos existentes en la pantalla a una segunda actividad que mostrará los datos de forma sencilla.



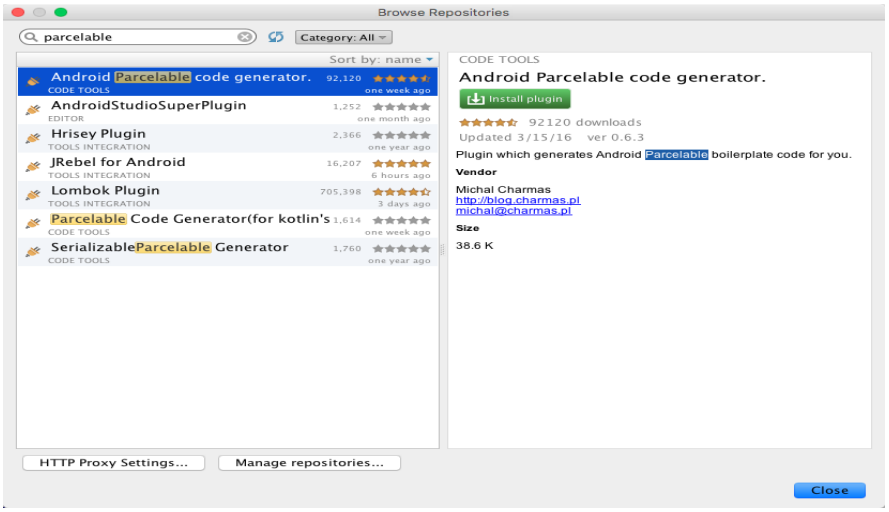
Creamos la clase Contacto:

```
package com.example.josebalmasedadelalamo.ejemploparcelable;

import android.graphics.Bitmap;
import android.os.Parcelable;

/**
 * Created by josebalmasedadelalamo on 30/9/17.
 */

public class Contacto{
    String nombre;
    String telefono;
    String email;
    Bitmap foto;

    public Contacto(String nombre, String telefono, String email, Bitmap foto) {
        this.nombre = nombre;
        this.telefono = telefono;
        this.email = email;
        this.foto = foto;
    }
}
```

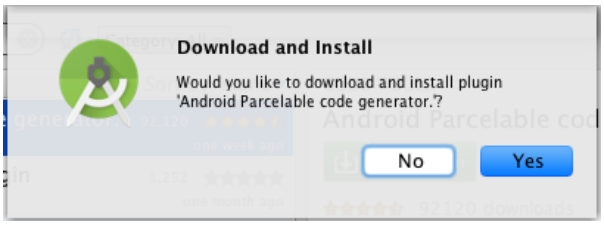
Para instalar el plugin debemos ingresar en Android Studio y buscar “Plugins”



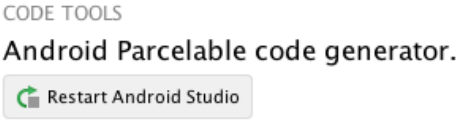
Ingresamos en el criterio de búsqueda “Parcelable” y damos click en instalar plugin



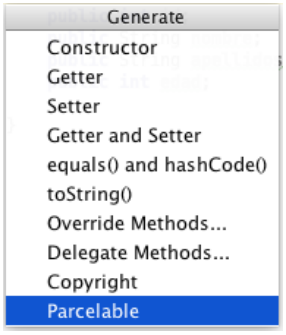
Nos solicitará confirmar si deseamos instalar el plugin y daremos click en “Si”



Una vez que se ha instalado debemos reiniciar Android Studio para que los cambios surtan efecto

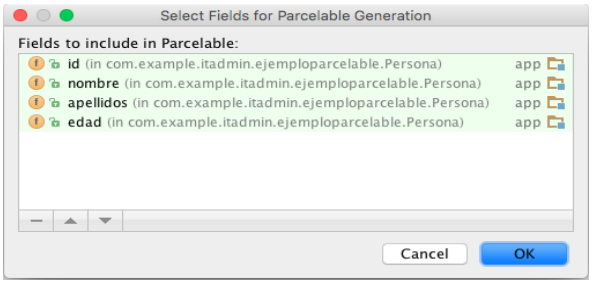


Ahora vamos a abrir nuestro proyecto en Android Studio y nos vamos a dirigir a una clase POJO creada anteriormente (Contacto). Damos click derecho dentro de la clase y seleccionamos la opción “Generar”



Aparece un nuevo menú donde elegiremos “Parcelable”

En la nueva ventana seleccionaremos los atributos a incluir para Parcelable, en nuestro caso deseamos pasar todo el objeto a otra Activity por lo que seleccionamos todos las propiedades



Después de unos segundos aparece en nuestra clase los métodos que se han generado. Podemos añadir al POJO los métodos Getters y Setters. Veamos ahora su uso.

En la Activity principal cuando se pulsa el botón hay que enviar la información a la activity

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    nombre=(TextView)findViewById(R.id.campo_nombre);
    tlf=(TextView)findViewById(R.id.campo_tlf);
    correo=(TextView)findViewById(R.id.campo_correo);
    img=(ImageButton)findViewById(R.id.placeImage);
    //imagen=((BitmapDrawable) img.getDrawable()).getBitmap();;
    imagen = BitmapFactory.decodeResource(getResources(),R.drawable.noimagen);
    Button button=(Button)findViewById(R.id.boton);
    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Contacto c=new Contacto(nombre.getText().toString(),tlf.getText().toString(),
                    correo.getText().toString(),imagen);
            Intent intent=new Intent (MainActivity.this,Activity2.class);
            intent.putExtra("Contacto",c);
            startActivity(intent);
            }
        });
}
```

secundaria. Solamente tenemos que crear el intent y pasarle el objeto.

La recuperación del objeto en la activity secundaria es igual de sencilla:

```
public class Activity2 extends AppCompatActivity {
    TextView nombre,tlf,correo;
    ImageButton img;
    Bitmap imagen;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_secundary);

        //recuperar datos y visualizar
        Contacto c;
        c=getIntent().getParcelableExtra("Contacto");
        nombre=(TextView)findViewById(R.id.txt_nombre);
        tlf=(TextView)findViewById(R.id.txt_tlf);
        correo=(TextView)findViewById(R.id.txt_correo);
        img=(ImageButton)findViewById(R.id.placeImage);

        nombre.setText(c.getNombre());
        tlf.setText(c.getTelefono());
        correo.setText(c.getEmail());
        img.setImageBitmap(c.getFoto());
    }
}
```